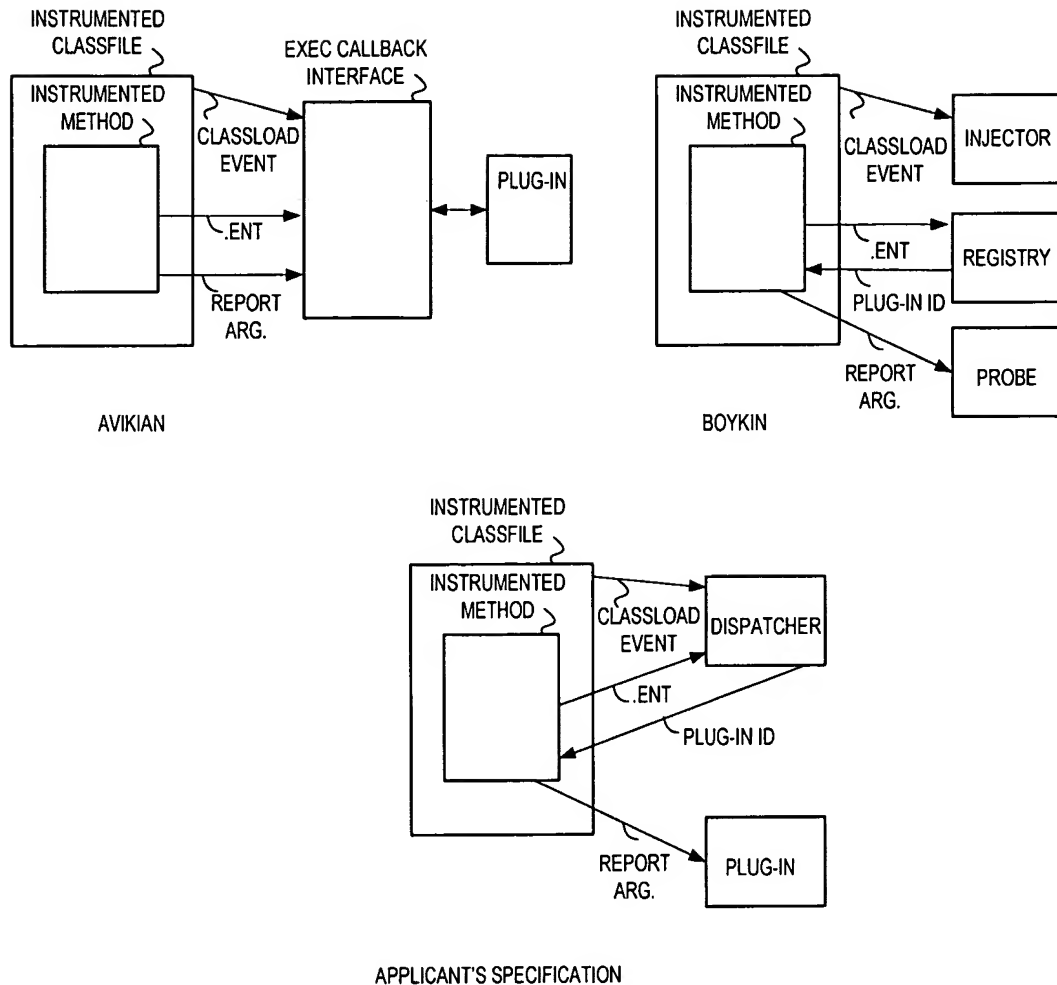


## COMMENTS

The enclosed is responsive to the Examiner's Final Office Action mailed 10/16/07 and is being filed pursuant to a Request for Continued Examination (RCE). In response, the Applicant has amended claims 1, 11, 13, 15, 17, 18, 28, 30, 31 and 34 and has canceled claims 35-43. The Applicant respectfully requests reconsideration of the present application and the allowance of the claims now presented.

### Examiner's Substantive Rejections

The Examiner has rejected independent claims 1 and 18 under 35 USC 103 primarily in view of the Avakian reference (U.S. Pat. Pub. No. 2005/0039171). The Applicant presents immediately below a visual summary depicting the more pertinent prior art references the applicant is presently aware of (Avakian and Boykin (U.S. Pat. Pub. No. 2004/0123279)) contrasted against the teachings of the present application's specification.



Essentially, the Applicant's specification describes a bytecode modification architecture that is fundamentally different than the architectures taught by Avakian or Boykin. As observed in the above figure, unlike the architectures disclosed by Avakian or Boykin, the Applicant's specification describes an architecture where an instrumented method and corresponding classfile invoke the same component (the dispatcher) upon class loading (depicted in the above figure as "CLASSLOADING EVENT") and method entry or exit (depicted simplistically in the above figure as ".ENT") but does not invoke that same component when reporting an argument to

the plug-in/probe monitor (depicted simplistically in the above figure as "REPORT ARG."). See, e.g., Applicant's specification, Fig. 18 and associated discussion describing a classfile 1801 that registers with the dispatcher 1802; *id.*, Fig. 17 and associated discussion, including para. [0146], which indicates registration with the dispatcher can take place during loading of the classfile; *id.*, Figs. 5a, 5b and associated discussion (describing invocation of the dispatcher by an instrumented method on method entry (Fig. 5a) or method exit (Fig. 5b) and showing invocation of the plug-in directly from the instrumented method (Figs. 5a, 5b)).

By contrast, Avakian's architecture invokes the same component (the ExecCallback interface) for all three events (i.e., classloading, method entry/exit and argument reporting), while, Boykin's architecture does not invoke the same component for both classloading and method entry/exit. See, e.g., Avakian, Fig. 2 (showing loaded instrumented class " C' " tied to a single ExecCallback interface 36); *id.*, Fig. 7 (showing all instrumentation method calls 702, 708, 716, 706 to the same interface "\$BIP\$hook"); *id.*, para [0064] (explicitly stating that the ExecCallback interface 36 is used so that plug-in 27A can receive "class load" events, method exit/entry events and method arguments); *id.*, para. [0066] (explicitly stating that the ExecCallback interface 36 is used so that plug-in 27A can receive "class load" events ("classLoadStart"), method exit/entry events ("methodEntry" and "methodExit") and method arguments ("reportArg")); *id.*, Fig. 9 and paras. [0107 - 00112] (describing the ExecCallback interface's "classLoadStart" and "classLoadEnd" methods); *id.*, Fig. 11 and paras. [0113-0114, 0117] (describing the ExecCallback interface's "methodEntry" and "methodExit" methods); *id.*, Fig. 10 and paras. [0115 - 0116] (describing the ExecCallback interface's "reportArg" method).

See, e.g., Boykin, Fig 4B (showing the injector 410 separate and distinct from the registry 416); id., paras. [0045], [0046], [0050] (stating explicitly that class loading events are passed to the injector); para. [0046] (stating explicitly that an instrumented method calls upon the registry to obtain the probe identification and then calls upon ("executes") the identified probe, "when a hook is invoked, the hook can determine whether a probe is enabled for its location by querying the registry. . . . If the registry has an enabled probe, then the hook gets and executes probe 418"); para. 0052 (stating explicitly that an instrumented method calls upon the registry to obtain the probe identification and then executes the identified probe).

The Applicant believes the distinction is patentable. More specifically, as described immediately below, the Applicant's approach is believed to be more versatile than the approaches taught by Avakian or Boykin.

Firstly, the Applicant's approach is more easily "re-configured" than Avakian because of the Applicant's use of a dispatcher (the use of the dispatcher being the reason why .entry/.exit events are directed to a different entity than the entity to which arguments are reported (the plug-in)). If a user of the Applicant's invention seeks to "change" the testing services applied to a particular method (e.g., by changing the applicable plug-in), the user need only update the code that controls the contents of the dispatcher's dictionary. The method's modified bytecode does not need to be recompiled nor do any plug-in/interface relationships need to be re-constructed.

By contrast, because Avakian ties an instrumented method to a particular interface and ties the called upon interface to a particular plug-in (or perhaps set of plug-ins), a change to the testing strategy would seem to require recompilation and modification of the method's byte code (to call out a different interface and/or new interface) and/or a restructuring of the relationship(s) that exist between existing interface(s) and their associated plug-in(s). Thus, to summarize, implementation of a change in testing strategy with the Applicant's approach merely involves updating a dictionary or look-up table, whereas, implementation of a change in testing strategy with Avakian's approach requires re-compilation of the code and/or re-defining the plug-in interface structure.

With respect to Boykin, Boykin only contemplates byte code modification during class loading. See, e.g., Boykin, para. [0034]. Hence, Boykin uses an "injector" which injects instrumentation hooks into a method's bytecode at class-loading. Boykin's injector is a wholly different entity than Boykin's registry which identifies the appropriate probe for an instrumented method. The Applicant's architecture, unlike Boykin, is capable of using bytecode that has been instrumented prior to class loading or during class loading - so, again - the Applicant's architecture is more versatile. Said another way, the Applicant's architecture can "work" with or without the presence of an entity that injects instrumentation bytecode during class loading. Here, the Applicant's dispatcher is notified of the class loading event (through a registration process) as a trigger for updating its internal dictionary in response to the presence of the newly loaded classfile. The Boykin reference simply fails to disclose this process (that is, Boykin's modified classfile does not register with the registry during classloading).

Applicant has amended the present claims to elaborate on these differences.

Specifically, claim 1 now recites (emphasis):

in an object oriented run-time environment, after a classfile has been loaded:

- a) invoking a second method from a first method, said first method belonging to said class, said invoking comprising providing an identification of said first method and said classfile, said second method belonging to a component, said classfile having previously registered with said component;
- b) said component performing said second method to identify a plug-in module for said first method based upon said identification, said plug-in module to implement a handler method, said component returning to said first method an identifier of said plug-in;
- c) invoking said plug-in module from said first method to execute said handler method to report and/or record information about said first method, said information including at least an argument included in said first method;
- d) executing said first method from a point beyond where said second method was invoked;
- e) flowing from said first method to a third method;
- f) invoking said second method from said third method, said invoking including providing an identification of said third method and a second classfile that said third method is a part of, said second classfile having been loaded at least by the completion of e) above, said second classfile having previously registered with said component;
- g) said component performing said second method to identify said plug-in module for said third method based upon said third method and second classfile identification, said component returning to said third method an identifier of said plug-in;
- h) invoking said plug-in module from said third method to execute said handler method to report and/or record information about said third method, said information including at least an argument included in said third method; and,
- i) executing a portion of said third method from a point beyond where said second method was invoked from said third method.

Here, in view of the figure provided above, the "first method" can be viewed as the instrumented method, the "second method" can viewed as the dispatcher method that is called (for instance) during entry or exit of the instrumented method, the dispatcher can be viewed as an embodiment of the "component", the "third method" can be viewed as another instrumented method, etc.

Thus the Applicant respectfully submits that the present application is in condition for allowance and respectfully requests the allowance of same.

Because the Applicant has demonstrated the patentability of all pending independent claims, the Applicant respectfully submits that all pending claims are allowable. The Applicant's silence with respect to the dependent claims should not be construed as an admission by the Applicant that the Applicant is complicit with the Examiner's rejection of these claims. Because the Applicant has demonstrated the patentability of the independent claims, the Applicant need not substantively address the theories of rejection applied to the dependent claims. Moreover, where the Applicant has failed to address a specific independent claim element alleged by the Examiner to be covered by prior art, such failure should not be viewed as an admission by the Applicant that the Applicant accepts or agrees with the Examiner's reasoning.

In the further interests of efficiency, the Applicant reserves the right under MPEP 2144.03.C to cause the Examiner to find in the prior art subject matter to which the Examiner has taken Official Notice at a later time in the prosecution of the present case when the subject matter of such prior art is actually at issue.

## CONCLUSION

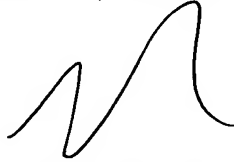
Applicant respectfully submits that the rejections have been overcome by the amendments and remarks, and that the pending claims are in condition for allowance. Accordingly, Applicant respectfully requests the rejections be withdrawn and the pending claims be allowed.

If there are any additional charges, please charge Deposit Account No. 02-2666 for any fee deficiency that may be due.

Respectfully submitted,

BLAKELY, SOKOLOFF, TAYLOR & ZAFMAN LLP

Date: 12/10/07



Robert B. O'Rourke  
Reg. No. 46,972

1279 Oakmead Parkway  
Sunnyvale, California 94085  
(408) 720-8300